Terminology

**System** - All files developed for the purposes of satisfying the client level requirements.

- **Data display module** - displays status updates and events to the user
- **Controller simulator** - Generates status updates and events then sends via RS422 serial protocol to the data display module.
- **Event Log file** - Will be generated by the data display module to contain 1 or more event strings encountered during a session.
- **Installer** - will deploy the project in the customers system and perform environment setup and initialization. Will record initial user preferences and take them into account during installation. I.e. “Do you want a shortcut on your desktop?”

**Environment** - The directory our system will be placed in and all of its contents.

**Status information** - (See class diagram for specifics) General data pertaining to the weapon which can be measured at any point in time during session.

**Status Update -** serialized version of status information to be sent to the data display module from the controller simulator.

**Event String** - A string of the format “<time> <event message> <param 1> <param 2> <param 3>” generated by the controller simulator to simulate the occurrence of a weapon related event. The parameters can be NULL, but the event message must be specific text outlining what the event is. Ex. “[00:12:41] Weapon overheat  237 200” where 237 represents measured barrel temp and 200 represents max recommended barrel temp in degrees celsius.

**Event update -** The serialized version of the event string to be sent to the data display module from the controller simulator.

**Session -** The time measured from the moment the controller sim is connected to the data display module to the moment the controller sim is disconnected from the data display module.

Client level requirements

- **REQ01** - The data display module shall be a desktop application.
- **REQ02** - The data display module shall read input data via RS422 serial protocol from the controller simulator.
- **REQ03** - The data display module shall have the ability to write the following data into a log file…
    a. All event strings recorded up to the point in which the user requests the event log file. (this is the case for generating a log file during a session)
    b. All event strings recorded throughout a complete session. Up to 5 complete session log files are kept before overwrites occur. (this is the case for automatically generating a log file after a session)
- **REQ04** - The data display module shall display all status information directly to the application's window for the duration of a session.
- **REQ05** - The controller simulator shall send event updates to the data display.
- **REQ06** - The data display module shall not require admin rights to install, set up, or use.
- **REQ07** - The data display module shall include filtering options to filter events and errors along the following categories
    a. show only errors
    b. show only cleared errors
    c. show only active errors
    d. show only events
- **REQ09** - The system and its environment shall be installed via an installer file.

System Level Requirements

- **REQ08** - The system and its environment shall be installed via an installer file.
- **REQ09** - The system shall be portable on devices running windows 10 or 11.
- **REQ10G** - The system should be portable on Debian Linux distributions.
- **REQ11** - The system shall include a controller simulator which will generate and send status updates and events to the data display module for testing purposes.

Software Level Requirements

- **SR01 -** The functions serializeStatus() and serializeEvent() shall be capable of generating bit string versions of given status data and event data respectably.
- **SR02 -** The functions deserializeStatus() and deserializeEvent() shall be capable of generating the original status data and event data, given a serialized bit string.
- **SR03 -** The connect() function shall be capable of performing handshake protocols with another device running connect().
- **SR04 -** The function sendMessage() shall be able to transmit a serialized string through the serial port.
- **SR05 -** The function readMessage() shall be able to detect incoming serialized messages from the serial port and update the status class with new data.
- **SR06** - The function disableComs() shall be able to pause serial communication.
- **SR07 -** The function enableComs() shall be able to resume serial communication.
- **SR08 -** The function initializeEventLL() shall be capable of creating a head node for a linked list given an event string.
- **SR09** - The function initializeElectricalLL() shall be capable of creating a head node for a linked list given a set of electrical data. Then adding an unknown number of new nodes for subsequent electrical components.

- **SR10** - The function addEvent() shall be capable of adding a new event to the event linked list.
- **SR11 -** The function outputEventLog() shall be capable of writing a log file, given a linked list of event data.
- **SR12 -** The function outputElectricalLog() shall be capable of writing a log file, given a linked list of electrical component data.
- **SR13 -** The function randomizeEvents() shall be capable of generating random event strings out of a given set of events at random time intervals within a given range.
- **SR14 -** The function randomizeStatus() shall be capable of generating random status updates every 0.1 seconds.
- **SR15 -** The function customEvent() shall be capable of taking in user input data and generating an event string using this data.
- **SR16 -** The function customStatus() shall be capable of taking in user input data and updating the status data structure using this data.
- **SR17 -** The function getInput() shall be capable of listening for user input data and deciding what to do with it.
-